European Scientific Journal June 2015 edition vol.11, No.18 ISSN: 1857 – 7881 (Print) e - ISSN 1857- 7431

CLEAN DIGITAL GARBAGE

Hong Zhou, PhD
Joseph Manthey, PhD
Ekaterina Lioutikova, PhD
University of Saint Joseph, West Hartford, Connecticut, USA

Abstract

Along with the vast price drop in computer disk space per unit, computer users have become less concerned with their disk space usage. This contributes to the accumulation of digital garbage which represents the unneeded files left behind in an information system. The accumulation of large quantities of digital garbage can incur significant costs and can even interfere the operations on current meaningful data. This article addresses this issue of digital garbage by presenting an intelligent algorithm that analyzes the age, file access time stamp, location environment, ownership, and content correlation with other current files for a given file to compute a clean-index (c-index) which determines the necessity of a file for removal.

Keywords: Digital garbage, disk space, c-index

Introduction

Computer digital storage space is becoming cheaper and cheaper, which contributes to the accumulation of digital data. From 1981 to 2014, the cost of computer disk space per gigabyte has dropped nearly 10 million times (A history of storage cost (update), 2014). It was roughly estimated that the space cost per unit has been halved roughly every 14 months (A history of storage cost, 2009), which is at a rate faster than the increasing pace of computer computation power predicted by Moore's law (Schaller, 1997). Today, software developers and general computer users are much less concerned about computer disk space. This creates an issue of disk space waste because computer users are developing the habit of not deleting unneeded files.

In the era of "big data" when data accumulates exponentially, it is not difficult to realize that obsolete data is accumulating at a faster and faster speed. While large quantities of data brings many opportunities, it also brings many challenges (Gantz & Reinsel, 2012). One challenge is how to keep the data clean and free of garbage.

"Digital garbage" represents the unneeded digital files in an information system. When there is enough disk space to spare, the drawbacks of digital garbage is not a serious issue. In fact, few organizations or companies spend resources to systematically clean their digital garbage because the gain is very limited compared to the resources to be spent on cleaning. However, digital garbage will eventually become a serious issue because of several reasons:

Digital garbage accumulates fast and takes significant disk space

- Digital garbage interferes with the searching and processing of meaningful data
- Digital garbage may create security issues since some garbage may contain sensitive data

Digital garbage is a more serious issue to those companies that are outsourcing their data into cloud storage because these companies need to pay for the storage. It can be foreseen that digital garbage will become an issue in the future, and therefore how to remove digital garbage while minimizing mistaken removals will be an important consideration.

The cleaning of digital garbage can be best achieved by individual users since each user understands the most about her/his files. In many organizations users are requested to archive or clean their files periodically. Unfortunately whether such requests are honored by users is uncertain. Currently there are two primary approaches to clean unneeded files in an information system.

User Dependent

User dependent garbage removal is the most reliable, efficient and accurate approach in digital garbage cleaning. For example, given a company with 10,000 employees, if all users are active in cleaning their own files, the mistakes happening in the cleaning process would be the minimum as each user has the best knowledge of her/his own files. Such a process is very efficient, too. The drawback of this approach comes from the fact that many users won't take the lead in cleaning their unwanted files. In addition, what if a user leaves a company leaving behind a large number of files on a disk?

Administration Forced

The administrative team who is in charge of an information system can initiate the garbage cleaning process. The team can force users to clean or archive files that are of certain number of years old. When a user does not archive or clean certain files that need to be cleaned, the administration team steps in and removes those files. There are several disadvantages associated with this approach. The fundamental disadvantage is again that there are users who are not aware of or do not pay attention to the file-cleaning

messages from the administrative team and therefore their wanted files may be mistakenly deleted.

The worst case scenario happens when a critical user leaves a company leaving behind a large number of files. It is difficult to determine what files of this user should be deleted because there might be very valuable information in those files. This becomes even truer for companies that are developing cutting-edge technologies where a critical user's knowledge might be left in those files which may or may not need to be accessed in a later time. Over years, the number of such uncertain files can accumulate and require significant disk space, increase the maintenance costs, and may even interfere the operation of current files. In such a case, an algorithm that can assess such files for removal becomes valuable.

Intelligent Digital Garbage Cleaning Algorithm

The ideal scenario is to have an automatic process that can delete unneeded files accurately without any mistakes. To achieve 100% accuracy alone by the algorithm might not be feasible, but a sophisticated algorithm can certainly help achieve the desired effect. This article presents an intelligent algorithm that can automatically remove unneeded disk files based on a clean-index (c-index).

The c-index takes into consideration the following factors of a file:

1) age;
2) time passed since last access time stamp;
3) location;
4) owner;
5) content correlation between a file with other files.

It is straightforward to understand why the age and the time passed since last access time stamp of a file are two critical factors in determining the necessity for removal. For example, a rule can be set up such that a file that is 20 years old or more should be deleted, or a file that has not been accessed in the past 10 years can be removed. Basically, the age and time lapse are positively correlated with the need for removal.

Location Factor

If a file has not been accessed for a long time while it resides in an environment where the data access traffic is heavy, should this file be removed based on the rules of age and access time stamp? Though there is no study about how likely similar files are stored together, it has been a common practice for professional computer users to store related files together. Thus, the location factor has two folds of meaning:

- If the neighboring files or parent directory has been accessed, this given file may have valuable information and therefore its removal likelihood is decreased.
- If the neighboring files or parent directory has not been accessed, this given file may be losing its value and therefore its removal likelihood is increased.

Owner Factor

In some cases, the owner of a file is critical and the ownership determines if a file needs to be kept forever. For example, some files of a CEO of a company may need to be kept for a long time. Hence, the importance of the owner in an organization or company is negatively correlated with the need for removal.

Content Correlation

This is a sophisticated factor in this algorithm. This factor is used to minimize the wrongly removal of meaningful files when such files are determined to be removable by other factors. A scenario can help explain this factor.

Suppose file A is determined to be removable by other factors. The analysis found out that A has information or data related to lung cancer that is the current working topic of the company. Clearly, there are many currently active files that are addressing lung cancer, though none of them is referencing file A. In this case, the algorithm has to make a decision if it is possible for file A to be referenced or accessed since its content is relevant to a current topic in this company. Generally speaking, if the content of an old file has strong correlation with current topics, its need for removal decreases.

The Algorithm

The c-index is computed as the following:

c-index = $(k_1(age - L_1) + k_2(lapse - L_2) + k_3 * LF + k_4 * CC) * OF$, where

age = the age of a file since its creation

lapse = the time passed since the last access to the file

LF = location factor

CC = content correlation

OF = owner factor

L_1 = age limit. A file's age must be over L_1 to be considered removable

L_2 = lapse limit. Time passed since last access must be over L_2 for a file to be considered removable

k_1 = the weight coefficient for age in the computation of c-index

k_2 = the weight coefficient for lapse in c-index computation
k_3 = the weight coefficient for LF in the computation of c-index
k_4 = the weight coefficient for CC in the computation of c-index

The c-index is a measure of the necessity to remove a given file. The larger the c-index, the greater the need for a given file to be removed. Thus, both k_1 and k_2 are positive numbers. The location factor represents the location importance of a file, which means LF >= 0. Correspondingly, $k_3 < 0$. Similar to LF, CC >= 0, and therefore $k_4 < 0$.

The value of owner factor is between 0 and 1 (or between 0 and a number larger than 1), with 0 representing the highest importance of the owner. When OF = 0, c-index = 0, which means that the file cannot be removed at all.

Computation of LF

Computation of LF needs some additional explanation. Since the file structure is hierarchical in all the computer operating systems, LF of a file (for example, file A) is affected by files that are either 1) in the parent directory of A; 2) in the same directory of A; and 3) in the sub-directory below A. This is illustrated in Figure 1.

A simple way to compute LF can be:
$LF = \sum f$, where f is a file that has been accessed recently. This means that LF is simply the number of the files in the user's home directory that have been accessed recently.

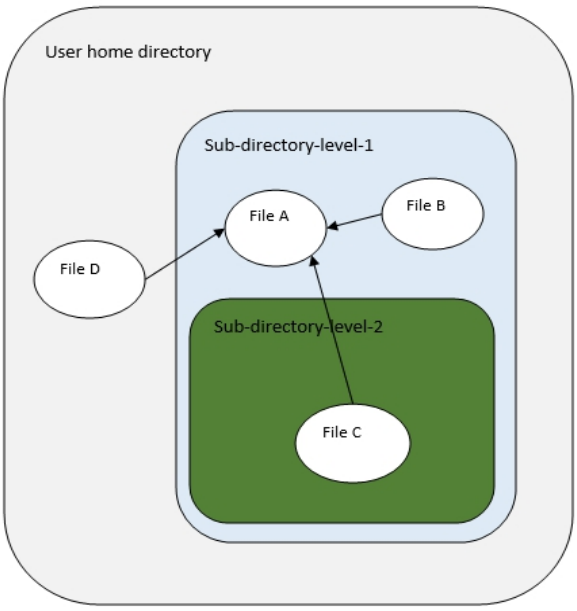

Figure 1. The LF of file A is determined by the file access stamps of file D which is in A's parent directory, file B which is in the same directory of A, and file C which is in the sub directory to where A resides.

A more sophisticated approach to LF computation might take into consideration of the distance between file A and other actively-accessed files. Using Figure 1 as one example where file B is in the same directory as A, while C and D reside in another directory directly above or below A. We can then simply compute LF as LF = (B) + 0.5 * (C) + 0.5 * (D). In this equation, a file inside a directory one level up or down is counted as only half compared to a file residing in the same directory as A.

Since there might be many files surrounding file A, the computation of LF for A might seem to be expensive. However, a tree approach can make the computation more efficient. For example, suppose that we are using the equation $LF = \sum f$ for LF computation. By using a tree approach, the LF values for all files inside the user's home directory can be computed in one traverse of the tree. This means that we do not compute LF for a single file alone. Instead, we compute the LF values for all files inside a user home directory. This can greatly reduce the time cost on LF computation.

Determination of Coefficients Ks

The c-index is determined by many factors in this proposed algorithm. The computation of LF, OF, CC, and the determination of L_1 and L_2 all require simulation and tests. The values of k_1, k_2, k_3, and k_4 are also unknown. The computation of c-index requires that we first determine the values of these 9 parameters.

The parameters of the algorithm can be represented as a vector of size 9. Once we have several information systems for use as a testing environment, we can use either pattern searching or another machine learning approach to approximate the best value ranges for the nine parameters. This process can be simplified if we assume that OF is a Boolean variable with a value of either 0 or 1. This leaves only 8 parameters to be determined. Furthermore, the values of CC and LF can be counter-adjusted by k_4 and k_3. Therefore, once the computational algorithms for CC and LF are finalized, what matters are the values of k_4 and k_3. Thus, the vector size is reduced to 6.

The values of L_1 and L_2 are usually system dependent, i.e. different organizations or companies that are applying this intelligent algorithm may have different values for L_1 and L_2. So, in the simulation and testing phase, we may need to find the best set of k_1, k_2, k_3, and k_4 for different L_1 and L_2 values. Generally speaking, L_1 and L_2 should be between 0 and 10, and are likely integers. Thus, the most critical parameters needing to be determined are k_1, k_2, k_3, and k_4.

Conclusion

This article presents an intelligent digital garbage cleaning algorithm which is based on the computation of c-index for each file. The value of c-index positively correlates with the need for removal. Even though such an algorithm may have no business place in today's information system environment where disk space is cheap enough to endure large amounts of digital garbage, such an algorithm may find its application in the near future when digital garbage accumulates to a degree such that it interferes with system performance or presents security concerns. Based on the design presented in this article, our future work will focus on the development of a software package that implements this algorithm. To help implement this algorithm, we hope to conduct several experiments to study how each file factor affects the need for removal of a file. This can help determine the coefficients k_1, k_2, k_3, and k_4, and improve the computation of LF. In addition, the authors would like to point out that what is presented in this article is just a preliminary design subject to improvements or modifications based on our further research discoveries.

References:

A history of storage cost (update). (2014, 3 9). Retrieved from http://www.mkomo.com/cost-per-gigabyte-update

A history of storage cost. (2009, 9 8). Retrieved from http://www.mkomo.com/cost-per-gigabyte

Gantz, J., & Reinsel, D. (2012). The Digital Universe in 2020: big data, bigger digital shadows, and biggest growth in the far east. In *IDC iView: IDC Analyze the Future.*

Schaller, R. R. (1997). Moore's law: past, present and future. *IEEE Spectrum*, 52-59.